На правах рукописи

Иссам Мустафа Ибрагим

ИНТЕГРИРОВАННАЯ СРЕДА РАЗРАБОТКИ ДЛЯ ИНФОРМАЦИОННО-РАСЧЁТНЫХ ЗАДАЧ

Специальность: 05.13.18 - математическое моделирование, численные методы и комплексы программ

А в т о р е ф е р а т
диссертации на соискание ученой степени
кандидата физико-математических наук

Казань - 2008

Работа выполнена на кафедре теоретической кибернетики, факультета ВМК, Казанского государственного университета им. Ульянова- Ленина.

Научный руководитель: кандидат физико-математических наук, доцент Еникеев Арслан Ильясович

Официальные оппоненты: доктор технических наук, профессор Латыпов Рустам Хафизович

кандидат физико-математических наук, доцент Гайфуллин Рашид Рахматуллович

Ведущая организация: Казанский государственный технический университет им. А.Н. Туполева

Защита диссертации состоится 6 марта 2008г. в 16 часов 30 мин. на заседании диссертационного совета Д. 212.081.21 при Казанском государственном университете им. В.И. Ульянова-Ленина по адресу: 420008, г. Казань, ул. Кремлевская, 18, корпус 2, аудитория 218.

С диссертацией можно ознакомится в научной библиотеке им. Н.И. Лобачевского Казанского государственного университета.

Автореферат разослан 5 февраля 2008г.

Учёный секретарь
Диссертационного совета,
д.ф.-м.н., доцент Задворнов Олег Анатольевич

Актуальность темы

Представляемая работа посвящена построению объектно-ориентированной модели программных систем, ориентированных на автоматизацию решения так называемых информационно – расчетных задач, в число которых включаются задачи компьютерной бухгалтерии, делопроизводства, статистики и т.п. Основной особенностью упомянутого класса задач является использование относительно простых структур данных, в большинстве случаев адекватно представляемых аппаратом реляционной алгебры. Однако, такое важное понятие, как иерархия абстракций, определяющее один из основных принципов объектно-ориентированного подхода, в рамках реляционной модели представляется далеко неадекватными способами, что может существенно отразиться на эффективности функционирования соответствующей интегрированной среды разработки программных приложений. Особенностью предлагаемого подхода является расширение реляционной модели путем включения аппарата фреймов с целью гибкого сочетания простоты реляционной модели со средствами эффективного представления иерархии абстракций. Основное предназначение включаемого фрейма – это построение специализированной интегрированной среды разработки, обеспечивающей возможность эффективного создания различных приложений для автоматизации решения информационно – расчетных задач, связанных с иерархической организацией данных и процессов их обработки. Поэтому в дальнейшем подобные фреймы будут называться фреймами разработки.

Одним из ключевых принципов создания подобной интегрированной среды, основанной на объектно-ориентированной модели, является возможность повторного использования различных компонент из разработанных ранее программных средств (так называемых паттернов проектирования). Объектно-ориентированный фрейм разработки реализуется в виде множества абстрактных классов, образующих функциональный базис, а также множества классов, специфичных для соответствующей предметной области.

Основу аппарата объектно-ориентированных фреймов разработки составляет набор гибких инструментальных средств, обеспечивающих удобство разработки и требуемую функциональную полноту создаваемых приложений для соответствующей проблемной области на основе иерархии абстракций. Иерархия абстракций предусматривает упорядочение абстракций путем их распределения по различным уровням, называемых уровнями абстракции. Уровни абстракций соответствуют степеням специализации фреймов разработки и позволяют разработчикам приложений проектировать свои специализированные фреймы разработки с целью эффективного создания различных требуемых приложений из соответствующей проблемной области. Объектно-ориентированные технологии представляют естественную среду для фреймов разработки. Аналогично тому, что одни классы могут представляться как экземпляры других более абстрактных классов, каждое приложение также можно

рассматривать как результат специализации некоторого абстрактного фрейма разработки. Использование фреймов разработки при создании приложений предусматривает построение специализированных классов или конкретных объектов для соответствующей проблемной области на основе множества абстрактных классов.

Цель работы

Основной целью диссертационной работы является создание методологии построения специализированной интегрированной программной среды, обеспечивающей увеличение производительности процессов проектирования, разработки и реализации приложений для автоматизации решения информационно-расчетных задач. Кроме этого предусматривается выбор формальной математической теории, которая позволила бы адекватно описать модель упомянутой выше специализированной интегрированной среды разработки, а также возможность проведения исследований свойств модели. Для программной реализации поставленных задач использована программная среда, основанная на средствах объектно-ориентированной СУБД *Visual FoxPro.*

Для достижения поставленной цели необходимо было решить следующие задачи:

1. изучить стандартные средства объектно-ориентированных баз данных;
2. изучить механизм так называемых паттернов проектирования (Design Patterns);
3. построить математическую модель для представления классов фрейма разработки, используемого для создания специализированной интегрированной программной среды, обеспечивающей увеличение производительности процессов создания программ автоматизации решения информационно-расчетных задач;
4. разработать множество абстрактных классов, образующих функциональный базис для соответствующей предметной области;
5. построить приложение в области информационно-расчётных задач на основе предложенного фрейма разработки с целью демонстрации и оценки адекватности предлагаемого множества абстрактных классов, образующих функциональный базис фрейма.

Методы исследования

При решении поставленных в диссертационной работе задач использован аппарат алгебраических спецификаций в сочетании с операциями теории категорий для формализованного описания внутренней структуры классов и отношений между классами.

Научная новизна результатов

1. Создание специализированной объектно-ориентированной модели и интегрированной программной среды на основе аппарата фреймов разработки с целью

повышения производительности процессов создания приложений из области информационно-расчетных задач.

2. Построение адекватных средств формализованного описания спецификаций классов фрейма разработки, обеспечивающих математическую строгость описания и возможность эффективной реализации методов автоматической генерации программного кода.

3. Разработка механизма безопасности, обеспечивающего сохранность данных и устранение возможности появления конфликтных ситуаций путем ограничения доступа пользователя в пределах выполняемого приложения.

Практическая ценность результатов

Расширение реляционной модели путем включения аппарата фреймов обеспечивает процесс эффективного построения и упорядочения иерархии абстракций при проектировании и разработке приложений из области информационно-расчетных задач. Уровни абстракций, получаемые при этом, соответствуют степеням специализации разрабатываемого приложения и позволяют разработчику адекватно выделять обобщающие аспекты различных частей приложения на каждом уровне. Такой подход уменьшает вероятность появления в разрабатываемом проекте дублирующих компонент и исключает в идеале необоснованную избыточность разрабатываемого проекта. С другой стороны, этот подход, предусматривающий адекватную структуризацию процесса разработки и проектирования приложения, оказывается полезным для повторного использования созданных при этом компонент в практической разработке других подобных приложений.

На защиту выносятся:

1. Объектно-ориентированная модель программных систем, основанная на фрейме разработки, определяемая в виде множества абстрактных классов, образующих функциональный базис, а также в виде множества классов, специфичных для соответствующей предметной области.
2. Формализованные средства для адекватного описания спецификаций классов, составляющих фрейм разработки.
3. Реализация механизма безопасности в виде совокупности модулей, предназначенных для обеспечения сохранности данных и устранения конфликтов между различными компонентами приложения в процессе его выполнения.
4. Демонстрационное приложение из области информационно-расчётных задач с целью проверки адекватности и эффективности построенного множества абстрактных классов, образующих функциональный базис фрейма разработки.

Апробация работы и публикации

Основные научные результаты диссертационной работы докладывались на следующих научно-технических конференциях:

- Международная молодёжная научная конференция, посвященная 1000-летию города Казани « Туполевские четения », Казань, Ноябрь 2005.
- VII Международная научно-практическая конференция « Общество, государство, личность: проблемы взаимодействия в условиях рыночной экономики », Апрель 2006.
- Итоговые научные конференции Казанского государственного университета в период 2005-2007.

Структура работы

Диссертационная работа изложена на 109 страницах текста, включающего в себя пять глав основного материала, 37 рисунков, 3 таблицы и список литературы, включающий 24 наименования.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Введение обосновывает актуальность темы, излагаются цели, задачи исследования, новизна и практическая ценность выносимых на защиту результатов.

Первая глава содержит обзор основных принципов организации объектно-ориентированных баз данных, включающих в себя вопросы однозначной идентификации, структурной организации, поведения, и инерционности объектов, а также механизм именования объектов. Далее приведен обзор стандартной объектной модели « ODMG 3.0 », включая принципы объектной модели, язык определения объектов (ODL), язык запросов (OQL), а также связь объектной модели « ODMG 3.0 » с языком программирования C++.

В той же главе представлен обзорный материал по так называемым паттернам проектирования (Design Patterns), включающий в себя такие методы, как шаблонный метод (Template Method), "посредник" (Mediator), фабричный метод (Factory method), абстрактный фабричный метод (Abstract Factory), делегирование (Bridge) и метод упаковки (Wrappers).

Далее предложено описание так называемых фреймов разработки (frameworks), в том числе их сферы применения, механизмы адаптации, их способы взаимодействия с создаваемыми приложениями, их изменяемые области (*hot spots*) и "замороженные" области, которые не подлежат изменениям (*frozen spots*), их способы взаимодействия с создаваемыми приложениями, и создание приложений на основе фреймов разработки, реализуемое с помощью так называемых хуков (Hook-функций). На рис.1 выделены следующие части фрейма разработки:

- ядро фрейма разработки включает в себя абстрактные классы, определяющие обобщенные аспекты структуры и поведения фрейма, и образует базис для разработки приложений на основе данного фрейма; кроме этого, в ядро могут быть включены конкретные классы для разработки приложений определенной предметной области;
- библиотека фрейма разработки представляет расширение ядра фрейма, содержащее конкретные компоненты, которые могут быть непосредственно использованы без дополнительной модификации, или, по крайней мере путем незначительной модификации в разработке приложений;
- надстройка для специализированных приложений содержит компоненты, определяемые полностью спецификой определенного приложения;
- приложение состоит из ядра, библиотеки фрейма разработки и надстройки;
- неиспользуемая часть библиотеки классов включает в себя множество классов из фрейма разработки, которые не будут использованы в данном приложении.

Рис.1. Структурная диаграмма разработки приложения на основе фрейма разработки

Кроме этого в данной главе представлены примеры фреймов разработки:
“ET++ SwapsManager”, “ITHACA” и “San Francisco”.

Во второй главе даётся описание алгебраического подхода, представляемого формальными средствами описания абстрактных структур, операциями и отношениями над ними на основе комбинирования алгебраических спецификаций и теории категорий. Суть таких моделей характеризуется следующими особенностями:

1) описание модели поведения системы представляется на основе алгебраических операций и аксиом;
2) сложные спецификации подлежат декомпозиции на более простые путем использования операций конструирования, дополненных операциями из теории категорий;
3) для описания абстрактных структур используется аппарат так называемой многосортной (многоосновной) алгебры, в которой каждому типу данных соответствует некоторый сорт (множество элементов, над которыми определены функции из сигнатуры алгебры) ;
4) функции в многосортной алгебре определяются сигнатурой, представляющей множество конструкций вида $f : s_1 \times s_2 \times \ldots \times s_n \rightarrow s$, где s_i ($i=1..n$) - сорта аргументов функции , s -сорт значения функции ; при $n=0$ функция вырождается в константу ;
5) алгебраическая спецификация формируется из сигнатуры и ассоциированных с сигнатурой аксиом.

Спецификация обеспечивает формальное описание внутренней структуры классов объектов, включающей описание атрибутов и операций . Однако , кроме этого требуется наличие средств для адекватного представления отношений между спецификациями , то есть более абстрактных или внешних структурных аспектов. К таким внешним аспектам относятся такие, как наследование, агрегирование, группировка посредством ассоциаций и т.п. Для описания упомянутых внешних структурных свойств можно использовать теорию категорий, представляющей адекватные средства для описания внешних структурных свойств различных формальных систем . Согласно определению категория *C* представляется в виде набора *C*-объектов таких , что :

1) для каждой пары *C*-объектов *X* и *Y* задано отображение $f: X \rightarrow Y$, называемое морфизмом и обозначаемое как $MOR_C(X,Y)$;
2) для пары морфизмов $f = MOR_C(X,Y)$ и $g = MOR_C(Y,Z)$ определена композиция $g * f = MOR_C(X,Z)$;
3) для каждого *C*-объекта *X* задан тождественный морфизм $id_X = MOR_C(X,X)$

причем выполняются две аксиомы :

A1) операция композиции ассоциативна : $h * (g * f) = (h * g) * f$;
A2) для любых $f = MOR_C(X,Y) : id_X * f = f$ и $g = MOR_C(Z,X) : g * id_X = g$

Отсюда, если функция *f* реализует морфизм, очевидно, что область определения функции *f* (обозначаемая как *dom(f)*) и область значений функции *f* (обозначаемая как *cod(f)*) представляются множествами *C*-объектов . Если a = dom(f) и b = cod(f) мы представляем это как:

$$f: a \rightarrow b$$

Каждое такое отображение определяет понятие *C*-отображения, а множество *C*-отображений образует морфизм.

Ориентированный граф, вершины которого определяют *C*-объекты, а дуги - *C*-отображения, называется диаграммой.

В качестве примера можно привести категорию *множество*, где каждое множество представляет в этом смысле *C*-объект, а в качестве *C*-отображений фигурируют функции и отношения над множествами. Другим примером является категория *спецификация*. В этом случае каждая из спецификаций рассматривается как *C*-объект, а в качестве *C*-отображений фигурируют отображения вида $f: < S, \Omega > ->< S', \Omega' >$, где сорта S одной спецификации отображаются в совместимые сорта S' другой, а сигнатура Ω - в сигнатуру Ω' другой, причем аксиомы первой спецификации являются теоремами второй спецификации .

Далее дается описание структуры классов с помощью аппарата алгебраических спецификаций: тип класса определяется сигнатурой $\Sigma = <S,\Omega>$ и множеством аксиом Ф над сигнатурой Σ, где :

S - множество сортов , включающих сорт класса ,

Ω - множество функций , заданных на множестве S .

Сорта из S используются для описания наборов значений данных в спецификации. В отличие от других сортов, сорт класса представляет множество всевозможных объектов класса, или, с точки зрения алгебраического подхода – множество всевозможных абстрактных представлений объектов класса .

Функции из множества Ω классифицируются как атрибуты, атрибуты состояния, события и операции. Атрибуты определяются непосредственно функциями, которые возвращают в качестве значений специальные элементы данных. Каждый атрибут определяется атрибутным описанием в классе *C* следующим образом:

имя атрибута: имя класса -> тип атрибута данных

Методы классов формально представляются как:

имя метода : имя класса , $p_1 \ldots p_n$ -> имя класса

где:

имя класса: представляет объект соответствующего класса,

$p_1 \ldots p_n$ - параметры метода.

Семантические описания функций определяются с помощью аксиом исчисления предикатов 1-ого порядка. В основном, аксиомы определяют методы посредством описания результатов их воздействия на значения атрибутов.

Далее даётся описание операций классов, которые обеспечивают различные преобразования на классах, но не могут непосредственно менять значения атрибутов. Семантику операций также можно определять с помощью аксиом исчисления предикатов первого порядка.

имя операции: имя класса, $p_1 \ldots p_n$-> тип операции данных.

В третьей главе приводится описание аппарата фрейма разработки, реализованного автором на основе объектно-ориентированной СУБД Visual

FoxPro. Представляемый аппарат фрейма разработки рассматривается здесь как расширение реляционной СУБД (в данном случае Visual FoxPro) с целью гибкого сочетания реляционных средств со средствами эффективного представления иерархии абстракций.

Создание приложений на основе фреймов разработки реализуется с помощью так называемых хуков (Hook функций), представляющих из себя специальные области фрейма, в которых осуществляется определение новых функций или переопределение старых в соответствии со спецификой приложения. Механизм хуков обеспечивает гибкость процессов построения, настройки и перенастройки приложений в пределах определенного класса задач. Фреймы разработки содержат в себе изменяемые области (*hot spots*) и "замороженные" области, не подлежащие изменениям (*frozen spots*). Изменяемая область может заполняться объектами и свойствами, специфичными для создаваемого с помощью фрейма разработки приложения. В то же время замороженные области содержат компоненты, являющиеся общими для соответствующего класса задач. Изменяемая область может содержать хуки, в замороженной области они не могут появиться. Фреймы разработки различаются сферой их применения и механизмами адаптации. По сфере применения различают универсальные и специализированные. Различают механизмы адаптации, построенные на принципах композиции ("черный ящик"), или на принципах наследования ("белый ящик").

Представляемый здесь фрейм разработки основан на принципах "белого ящика". В этих фреймах механизмы адаптации реализуются через изменяемые области, основными компонентами которых являются хуки, представляемые в виде шаблонных методов (template methods) и паттернов проектирования (design patterns).

Фрейм разработки определяется совокупностью абстрактных классов, которая обеспечивает общую инфраструктуру для приложений определенной предметной области. Пользователь фрейма разработки должен понять внутреннюю структуру фрейма разработки для того, чтобы использовать его для построения соответствующего приложения. Для описания структуры фрейма разработки, использованы UML диаграммы (диаграмма классов, объектов, и последовательности) и UML-F - совокупность стереотипов в форме Профиля UML, которая используется для определения модели фреймов разработки. Профиль UML представляет собой часть языка UML и расширяет его с помощью стереотипов, предназначенных для специальных целей. Стереотипы являются механизмом расширения ядра языка UML. Стереотипы записывается с помощью текста, заключенного в кавычки (например «framework») , однако они могут также изображаться с помощью пиктограммы стереотипа. В табл.1 представлены стереотипы UML диаграмм, которые используются для описания классов фрейма разработки.

На рис.2 представлена многоуровневая концептуальная модель фрейма разработки. Ниже приводится описание уровней этой модели и классов, включаемых в модель:

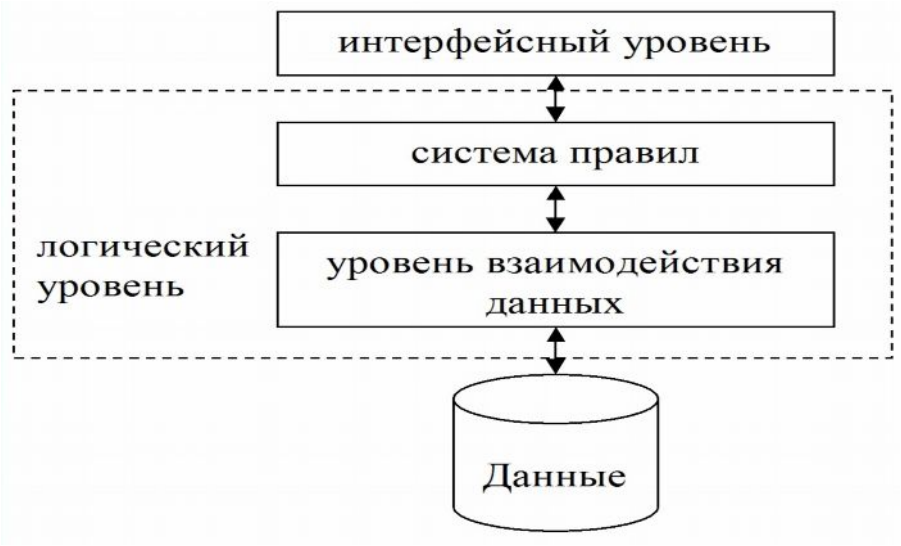

Рис.2. Концептуальная модель фреймов разработки

- *интерфейсный уровень* (*Presentation Layer*) – представляется средствами построения и управления экранными формами на основе классов “Aform”, “DataEntryForm”, “FWLoginForm” “FWButton”, “FWTXTBox”, “FWEditBox”, “ExitButton” и “DataEntryCMDButton” ;

- *уровень взаимодействия данных* (*Data Connection*) и *система правил (Business Rules)* вместе представляют единый логический уровень, используемый для представления так называемых Бизнес-Объектов (объектов используемой предметной области). Здесь используются классы “AdataEnvironment”,“FWDataEnvirronment”, “Acursor”, “FWCursor” ;

- *уровень хранения данных* (*Data*) представляет базы данных.

В зависимости от принадлежности к различным частям структурной диаграммы разработки приложения на основе фрейма разработки (см. рис.1), различают:

1. Классы, представляющие *ядро* фрейма разработки:
{ “DataEntryForm”, “FWDataEnvirronment”, “FWCursor”};

2. Классы, представляющие *библиотеку* фрейма разработки. Эти классы в свою очередь делятся на три группы:

А. Классы для конфигурирования среды:
{AEnvironment”, “DataSessionEnv”, “AppEnv”, “SaveEnv”},

Б. Классы, представляющие совокупности:
{“FWCollection”, “FWStack”, “FWSet”, “FWList”} и

В. Классы элементов управления для экранных форм: {“FWButton”,“FWTXTBox”,“FWEditBox”,“ExitButton”,“DataEntryCMD Button”, ...}.

Табл.1. Значение Стереотипов UML-F

Стереотип	Значение
...	Графическое представление класса содержит часть списка атрибутов и методов.
©	Графическое представление класса содержит полный список атрибутов и методов.
«unif-TH»	Класс содержит шаблонные методы (T) и методы зацепки (хуки- hooks (H)). адаптация класса будет осуществляться с помощью наследования.
«unif-t»	Шаблонный метод (Template Method).
«unif-h»	Хук метод (Hook Method).
«Framework»	Класс принадлежит фрейму разработки.
«Application»	Класс принадлежит приложению.
«Bridge-Abstraction»	Класс представляет абстракцию паттерна Моста (Bridge Pattern)
«Bridge-Implementation»	Класс представляет реализацию паттерна Моста (Bridge Pattern)
«Bridge-operation»	Метод принадлежит классу, представляющему абстракцию паттерна Моста (Bridge Pattern).
«Bridge-operationImp»	Метод принадлежит классу, представляющему реализацию паттерна Моста (Bridge Pattern) и используется для реализации метода класса, представляющего абстракцию Паттерна Моста (Bridge Pattern).
«Bridge-Imp»	Отношение между классами, реализующими Мост Паттерн (Bridge Pattern).
«adapt-static»	Класс используется во время разработки. Методы доопределяются в подклассах. Можно добавлять новые методы и атрибуты к подклассу, а существующие хук- методы могут быть доопределены в подклассах.

На рис.3 представлена диаграмма классов конфигурирования среды. Известно, что в СУБД Visual FoxPro специфика выполнения многих команд и операций зависит от конфигурации среды. Приведенный на рис.3 набор классов используются для сохранения, установки и восстановления

параметров конфигурации среды функционирования системы. Сюда включены следующие классы:

- *datasessionenv*;
- *appenv*;
- *saveenv;*
- *aenvironment*.

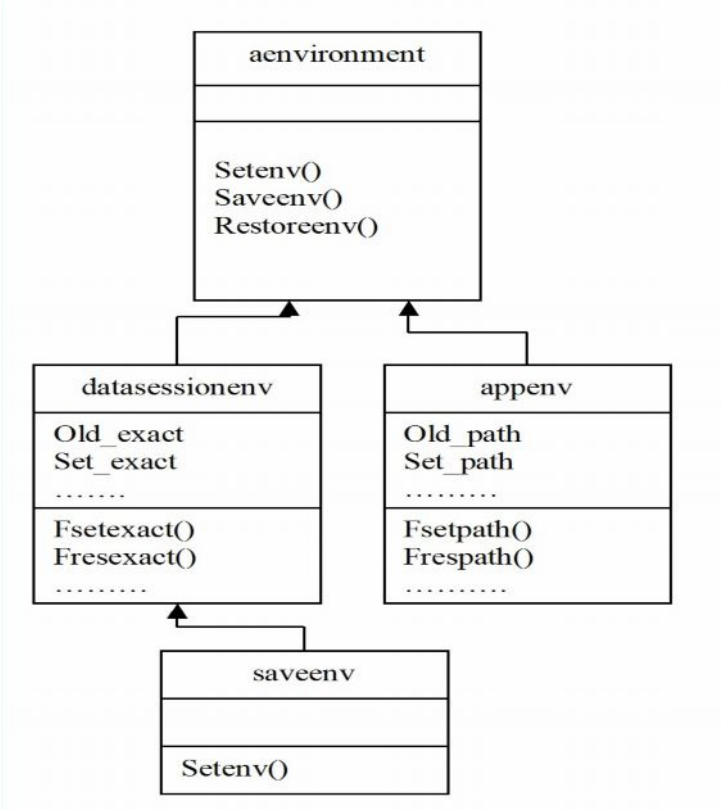

Рис.3. Классы конфигурирования среды для сессии данных и приложений

"DataSessionEnv" класс обеспечивает сохранение, установку и восстановление параметров конфигурации соответствующей сессии данных (*date, decimal, deleted, exact,…*). "*AppEnv"* класс «отвечает» за сохранение, установку и восстановление параметров для переменных в приложениях (*path, procedure, classlib,…*). *"SaveEnv"* класс, являющийся подклассом класса *"DataSessionEnv"*, обеспечивает сохранение состояния до вызова некоторого процесса с последующим восстановлением исходного состояния после выхода из процесса. "*Aenvironment"* класс – суперкласс, объединяющий все перечисленные выше классы в качестве подклассов. Этот класс играет роль буфера между классами конфигурирования среды фрейма разработки и классами СУБД Visual FoxPro.

Далее дается описание основных классов, входящих в концептуальную модель фрейма разработки:

- "*DataEntryForm*" класс используется для представления интерфейсного уровня. Этот класс наследует свойства от класса "*AForm*". Класс "*Aform*" играет роль буфера между классами экранных форм Visual FoxPro и классами

фрейма разработки, представляющими интерфейсный уровень. На рис.4 представлена диаграмма классов интерфейсного уровня, а в табл.1 дано описание UML стереотипов, использованных в диаграмме классов.

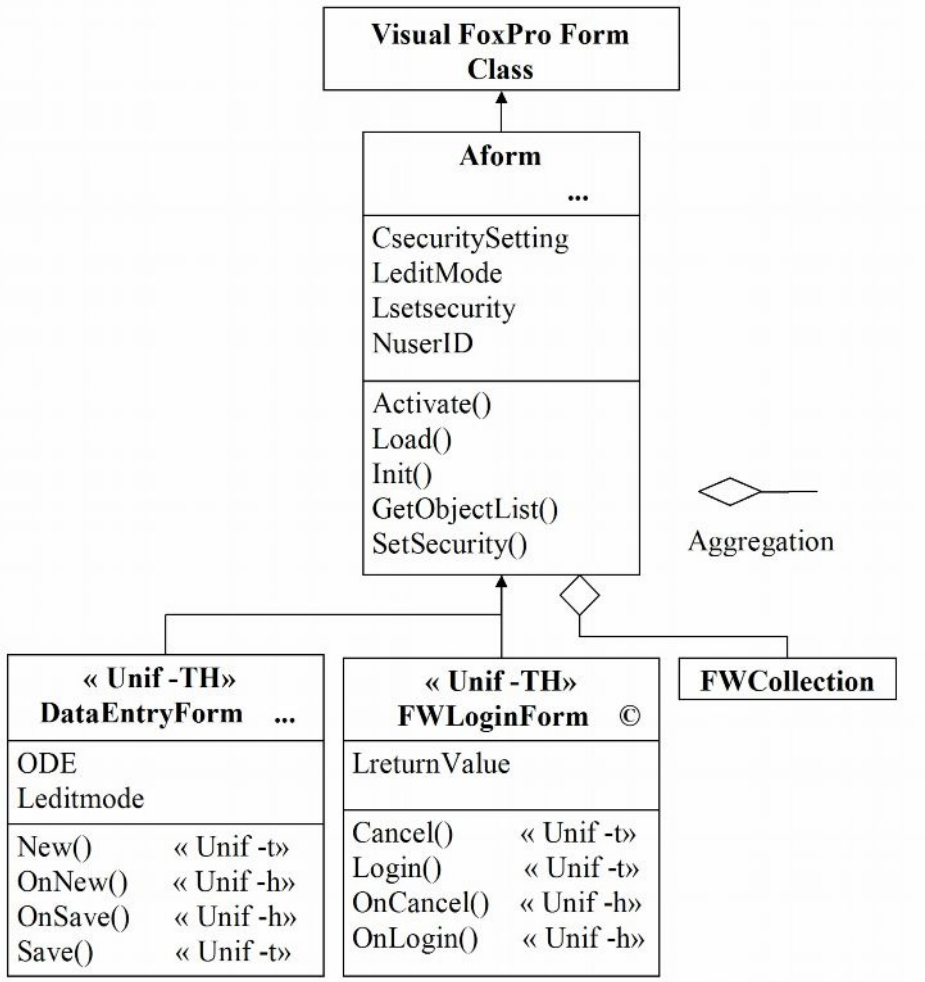

Рис.4. Диаграмма классов Интерфейсного уровня фреймов разработки

- Класс "*fwdataenvironment*" используется для представления так называемых бизнес-объектов. Класс бизнес-объектов предназначен для представления сущностей внешнего окружения разрабатываемых приложений. Класс бизнес-объекта имеет внутреннее и внешнее представления. Внутреннее представление этого класса, определяется информацией о сущности реального окружения, а его внешнее представление определяется через набор атрибутов и функций, связанных с сущностью реального окружения. "*fwdataenvironment*" представляет связь между интерфейсным уровнем и базой данных. Разработчик приложения определяет бизнес-правила внутри этого класса. Бизнес-правила используются для определения истинности информации о сущности реального окружения. На рис.5 представлена диаграмма классов "*fwdataenvironment*".

Объекты класса “*fwdataenvironment*” играют роль контейнеров объектов класса “*fwcursor*”.

- Класс “*adataenvironment*” играет роль буфера между классом Visual FoxPro и классами фрейма разработки, представляющими логический уровень.

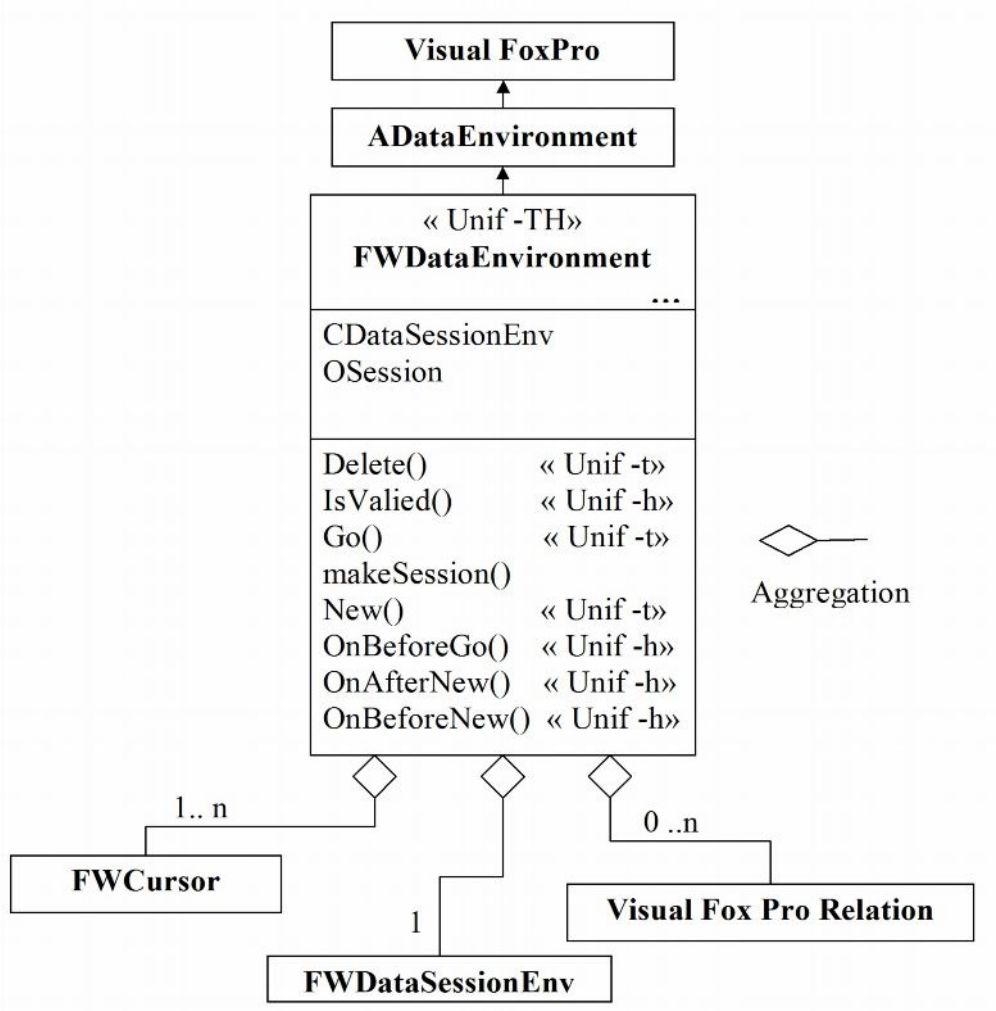

Рис.5. Диаграмма классов “*FWDataEnvironment*”

-Класс “*FWCursor*” представляет уровень взаимодействия данных и включен с целью распространения принципов объектно-ориентированного программирования на так называемые курсор-объекты СУБД Visual FoxPro, представляющие ссылки на таблицы баз данных. Известно, что аппарат обработки курсор-объектов в СУБД Visual FoxPro ограничен процедурной ориентацией.

Класс “*FWCursor*” наследует свойства от Класса “*ACursor*”. Класс “*ACursor*” непосредственно наследует свойства класса “*Cursor*” в системе Visual FoxPro. Этот класс играет роль буфера между классами Visual Fox Pro и классами фрейма разработки, представляющими уровень взаимодействия данных. На рис.6 предоставлена диаграмма этого класса.

Далее приводится описание отношений между классами фрейма разработки. На рис.7 дается описание отношений между интерфейсным классом "*DataEntryForm*" и классом логического уровня "*Fwdataenvironment*".

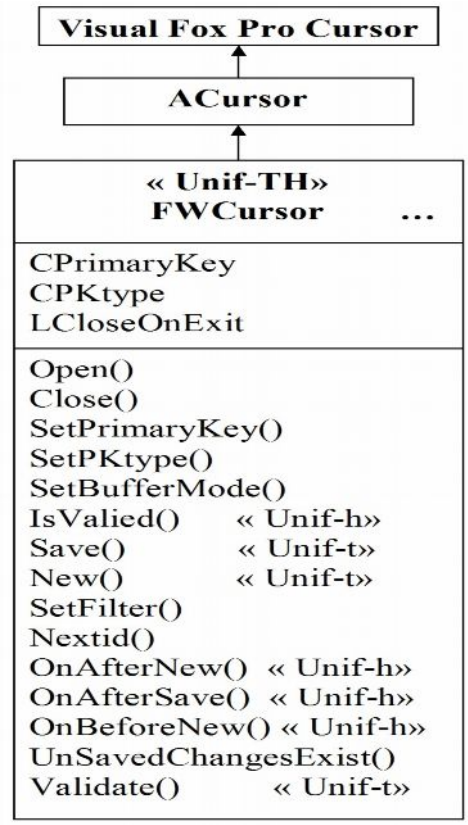

Рис.6. Диаграмма классов "*Fwcursor*"

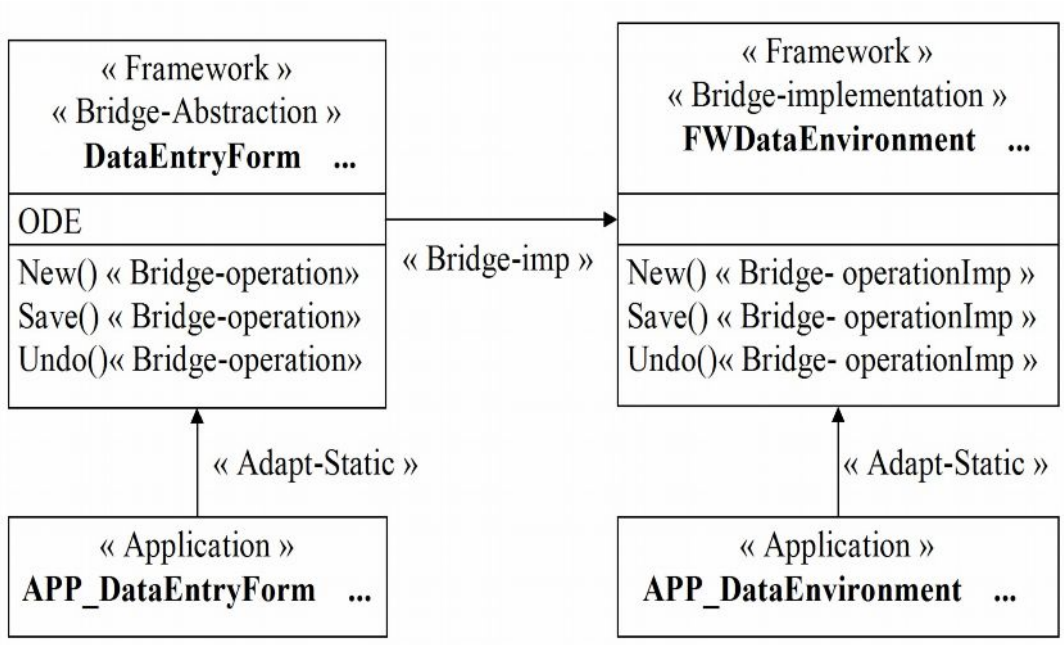

Рис.7. Отношения между классами фрейма разработки

Отношения между этими классами реализованы путем использования так называемого «Моста» паттерна проектирования (Bridge pattern). “*DataEntryForm*” абстрагирует «Моста» паттерн, а “*FWDataEnvironment*” выполняет реализацию этого паттерна. «Мост» паттерн используется для обеспечения гибкости фрейма разработки. С помощью «Мост» паттерна обеспечивается относительная независимость между классами интерфейсного уровня и классами логического уровня. Таким образом классы, принадлежащие этим уровням, могут изменяться вне зависимости друг от друга.

Далее дается описание основных операций фрейма разработки с помощью UML диаграмм последовательности. Например, на рис.8 представлено описание сценария операции “save”, которая используется для сохранения ожидающих редактирование_ (pending edits) всех таблиц одного бизнес-объекта. На рис.8 определены хуки, с помощью которых, пользователь фрейма разработки может изменить поведение операции с целью адаптации к соответствующим требованиям. Суть этой операции сводится к следующему:

1. На диаграмме последовательности объекты классов (“*DataEntryForm*”, “*FWDataEnvironment*”, и “*FWCursor*”) изображаются прямоугольниками на вершине вертикальной пунктирной линии. Эта вертикальная линия представляет собой жизненный цикл объекта в процессе взаимодействия.
2. Сообщения представляются стрелками между линиями жизни объектов (например **Save() сообщение**). Порядок следования сообщений устанавливается сверху вниз. **Рекурсивные вызовы** – сообщения, которые объекты посылают самим себе (например **Valid()**). При передаче такого сообщения стрелка указывает на ту же самую линию жизни.
3. Вводится изображение прямоугольника активности для показа периода времени, в течение которого объект является активным.
4. Управляющая информация представлена двумя способами. Первое – указывается **Условие** передачи сообщения (пример, **[**ODE exist**]**). Условие указывается в квадратных скобках. Сообщение посылается, тогда, когда это условие истинно. Второе - использование итерации при многократной передаче сообщения для множества принимающих объектов. Такая итерация указывается в квадратных скобках с предшествующей звёздочкой (на пример ***[for each cursor]**).
5. **Возвраты** изображаются пунктирной линией. Возврат указывает возврат от переданного ранее сообщения (например, **llok- Local Logical OK**).

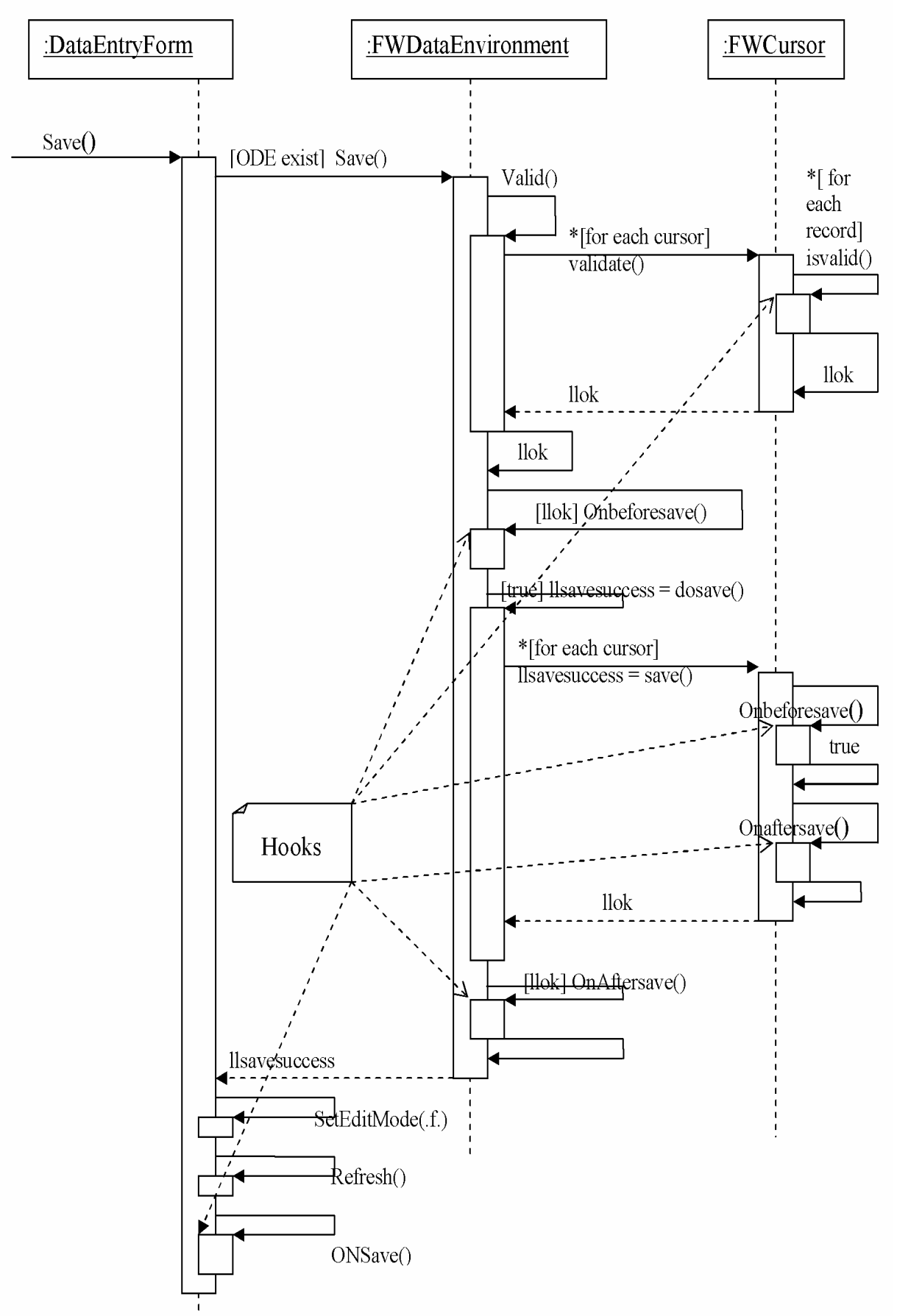

Рис.8. UML диаграмма описания операции “Save()”

В четвёртой главе предложено описание двух разработанных модулей. Это - модуль безопасности и модуль передачи сообщений для фрейма разработки. Свойства этих модулей наследуются из класса "*FWDataEnvironment*".

Модуль безопасности работает на основе спецификации приложения, включающего в себя определение пользователей, групп пользователей и списка логических секций приложения.

Каждый элемент управления в фреймовой разработке имеет "cSecuritySetting" свойство, которое ссылается на логическую секцию приложения. При передаче этого элемента управления модулю безопасности, значение его свойства "cSecuritySetting" определяет режим работы модуля безопасности. В табл.2,3 и на рис.9 дано описание работы модуля безопасности. На рис.10 представлена диаграмма описания объектов этого модуля.

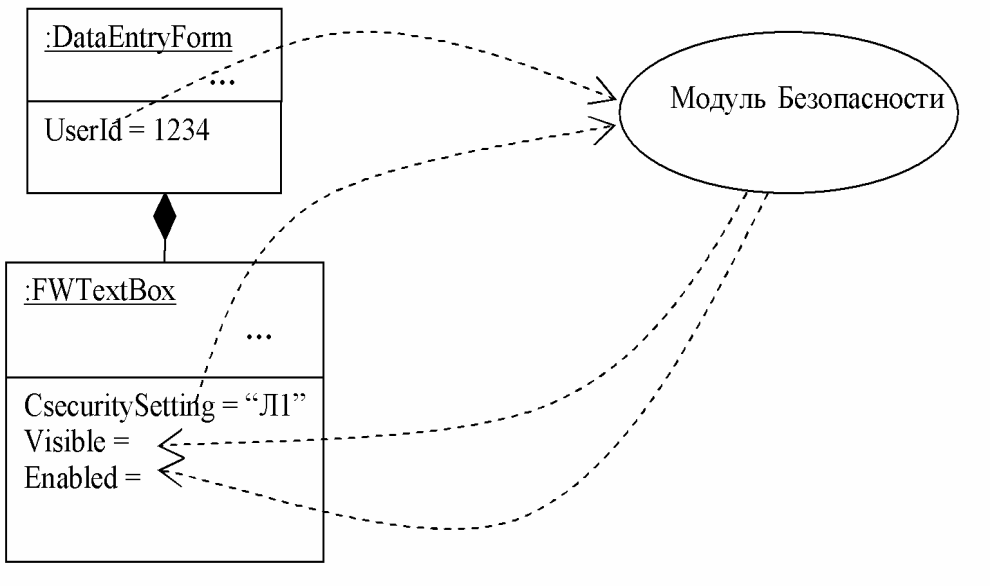

Рис.9. Пример использования модуля безопасности

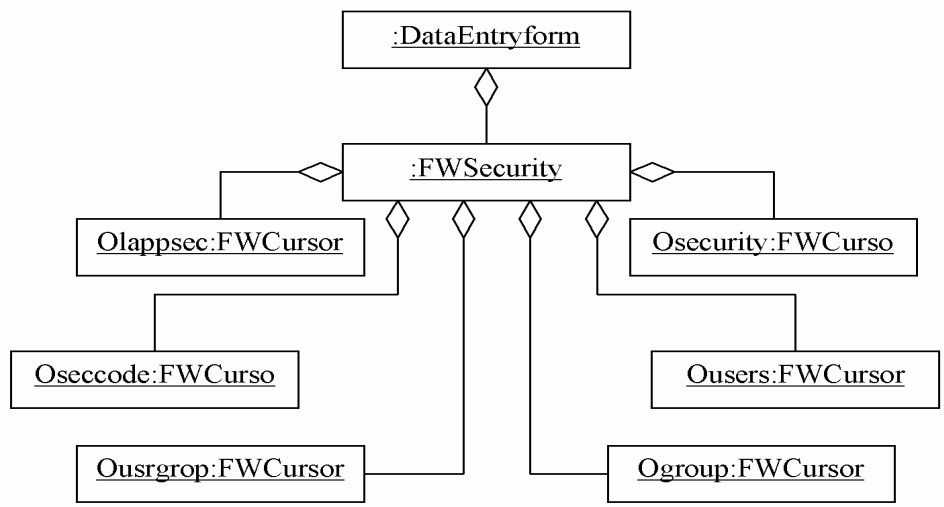

Рис. 10. Диаграмма Объектов модуля безопасности

Табл.2. Пример описания спецификации приложения.

		Группы Пользователей			
Логические секции приложения		Г1	Г2	Г3	...
	Л1	RW	…	…	…
	Л2	…	RO	…	…
	Л3	…	…	HID	…
	...	…	FORM	…	…

В примере из табл.2 представлены { Л1, Л2, Л3,...}логических секцией, набор групп пользователей { Г1, Г2, Г3,....} и описание кодов для установки соответствующего режима работы модуля безопасности (RO, RW,…).

Табл.3. Описание Кодов для модуля безопасности.

Установка	**Абревиатура**	**Свойства**
Чтение-Запись	RW	Enabled = .t. Visible = .t.
Только-Чтение	RO	Enabled = .f. Visible = .t.
Невидимый	HID	Visible = .f.
Форма	FORM	NA (*not allowed*)

Далее дается описание модуля передачи сообщений. Основное предназначение этого модуля заключается в возможности удобного изменения текстов, выдаваемых на экран из фрейма разработки, а также способов их вывода. Модуль передачи сообщений наследует свойства из класса “*FWDataenvironment*”. На рис.11 представлена диаграмма описания объектов модуля передачи сообщений.

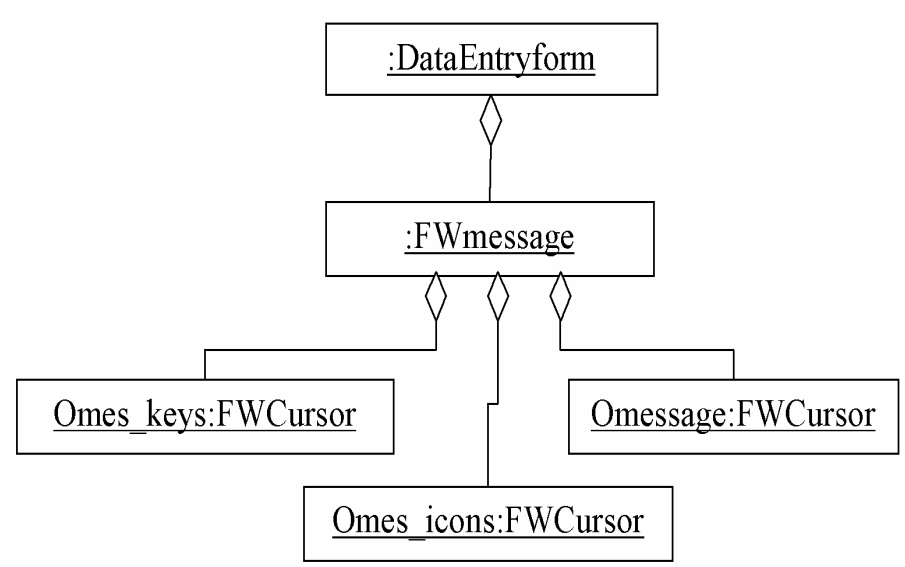

Рис.11. Диаграмма Объектов модуля службы сообщений

В пятой главе дано описание демонстрационного приложения, сгенерированного с помощью аппарата фрейма разработки (см. рис.12). Приложение демонстрируется на примере выписывания счетов клиенту. Приведенное приложение демонстрирует способ использования классов фрейма разработки в процессе создания приложения.

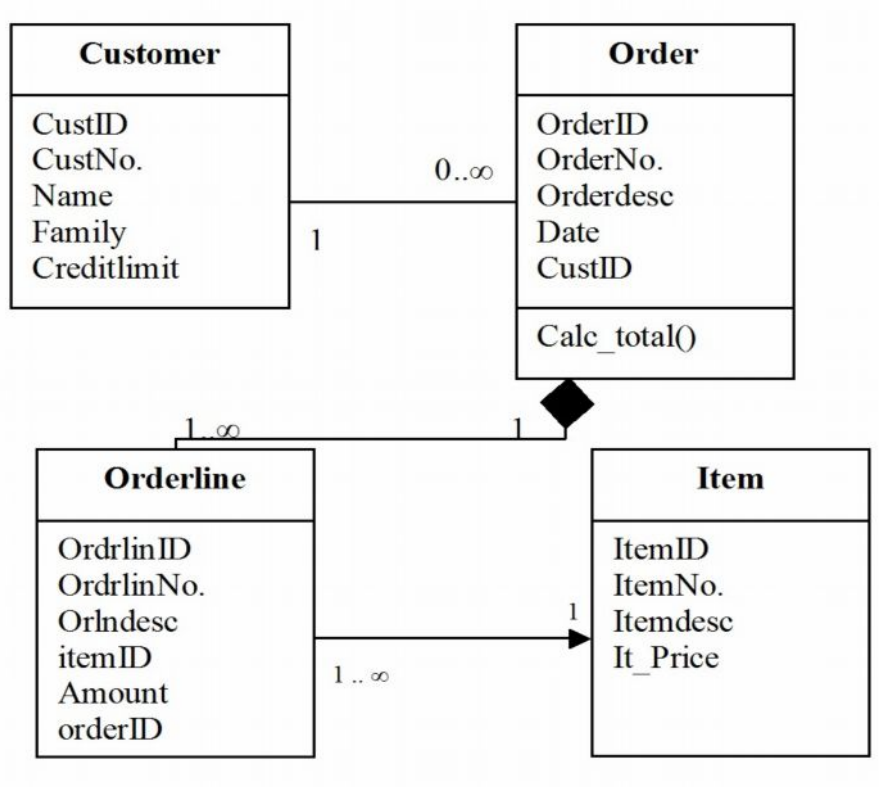

Рис.12. Диаграмма классов приложения демонстрационного примера

ОСНОВНЫЕ ВЫВОДЫ И РЕЗУЛЬТАТЫ

1. Построена объектно-ориентированная модель интегрированной программной среды, обеспечивающая эффективное создание приложений для автоматизации решения информационно – расчетных задач на основе аппарата фрейма разработки.

2. Предложены адекватные средства формализованного описания модели на основе аппарата алгебраических спецификаций вместе с операциями теории категорий для эффективного представления внутренней структуры классов объектов и отношений между ними.

3. На основе классов фрейма разработки спроектирована и выполнена программная реализация модуля безопасности и модуля передачи сообщений.

4. Создано приложение в области информационно-расчётных задач на основе аппарата фрейма разработки с целью оценки эффективности и демонстрации способов применения упомянутого аппарата.

ОСНОВНЫЕ ПУБЛИКАЦИИ ПО ТЕМЕ ДИССЕРТАЦИИ

1. Ибрагим И.М. Интегрированная среда разработки для информационно-расчётных задач / Ибрагим И.М. // Основные особенности применения объектно-ориентированных СУБД на современном этапе. Международная

молодёжная научная конференция, посвященная 1000-летию города Казани

«Туполевские чтения ». - Казань: Изд-во Казанского государственного

технического университета, Ноябрь 2005. - С. 27-28.

2. Ибрагим И.М. Интегрированная среда разработки для информационно-расчётных задач / Ибрагим И.М. // Один из подходов к созданию специализированной интегрированной среды разработки. VII Международная научно-практическая конференция « Общество, государство, личность: проблемы взаимодействия в условиях рыночной экономики ». - Казань: Издательский центр Академии управления «ТИСБИ», Апрель 2006. - С. 348-349.

3. Ибрагим И.М. Интегрированная среда разработки для информационно-расчётных задач / Ибрагим И.М., Еникеев А.И. // Принципы построения специализированной интегрированной среды разработки на основе фреймовых структур. – Казань, «Исследования по прикладной математике и информатике» КГУ, вып. 26, Изд-во КГУ , 2006. - С. 50-60.

4. Ибрагим И.М. Интегрированная среда разработки для информационно-расчётных задач / Ибрагим И.М, Еникеев А.И. // О Методологии создания специализированных объектно-ориентированных приложений на основе фреймовых структур. Исследования по информатике. Вып. 11. – Казань: Изд-во “ Отечество”, 2007. - С. 109-122.

5. Ибрагим И.М. Интегрированная среда разработки для информационно-расчётных задач / Ибрагим И.М. // Принципы разработки специализированных объектно-ориентированных приложений. Системы управления и информационные технологии. Научно-технический журнал № 1.1(27). – Воронеж: Изд-во «Научная книга», 2007. - С. 155-159.